

Notice of Allowability

Application No.

09/243,101

Examiner

Tuan A. Vu

Applicant(s)

SUSSER ET AL.

Art Unit

2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to 4/24/2006.
2. ☒ The allowed claim(s) is/are 59, 62-65, 67-73, 75-77, 80-96, 98, 102-105, 107-112, 116-119, 121-126, 130-133, 135-140, 144-147, 149-150 (renum 1-69).
3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 - a) ☐ All b) ☐ Some* c) ☐ None of the:
 1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.
 5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
 - (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
 - 1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
 - (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.
- Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO-1449 or PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application (PTO-152)
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date 7/20/06.
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 4/24/2006.

As indicated in Applicant's response, claims 59, 69, 77, 87, 95, 109, 123, and 137 have been amended. Claims 59-150 are pending in the office action.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Forrest Gunnison, Reg. # 32,899 on 7/20/2006.

The application has been amended as follows.

I) In the CLAIMS:

Claim 59:

~~An~~ A computer-implemented method for converting and transforming an application software program comprising: to generate an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions residing on a computer-readable medium, said instructions comprising operation codes and operands, said program sequence of instructions operable to be loaded to and executed by a resource-constrained device, said sequence of instructions previously converted from at least one class file, the method comprising:

(A) analyzing instructions from said at least one class file and any associated references to a constant pool and arithmetic operations having 32-bit operands said conversion; and

Art Unit: 2193

(B) removing said references to the constant pool and transforming said 32-bit operands based upon said analyzing by:

(i) transforming at least one reference, of at least one of said instructions, to information in a constant pool to data inlined by inlining directly in at least one additional operand to said at least one instruction, wherein said at least one additional operand specifies a data type; or opcode of said at least one of said instructions so that said at least one reference to information in said constant pool is eliminated because accessing said information in any constant pool is unnecessary for said at least one of said instructions;

(ii) replacing an operation code of another of said instructions, which accesses an entry in said constant pool to obtain a data type that was any one of a plurality of data types, with another operation code selected from a family of operation codes, wherein

(a) each said another operation code in said family is for a different one of said plurality of data types so that the data type is explicit for said each said another operation code in said family; and

(b) said another operation code represents said family for said entry in said constant pool; and

(iii) transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual

machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; and

wherein the instructions formed by replacement and transformation according to (i), (ii), (iii) are directly executable by the virtual machine on said resource-constrained device without any other intermediate stage of reconversion or referencing.

Claim 60: (Cancelled)

Claim 61: (Cancelled)

Claim 62:

The ~~method~~software program of claim 59 wherein said resource-constrained device is based on a 16-bit processor architecture.

Claim 63:

(Currently Amended) The ~~method~~software program of claim 59 wherein said resource-constrained device is based on an 8-bit processor architecture.

Claim 64:

The ~~method~~software program of claim 59 wherein said resource-constrained device comprises a random access memory with a capacity of no more than about 64 kilobytes.

Claim 65:

The ~~method~~software program of claim 59 wherein said resource-constrained device comprises a random access memory with a capacity of no more than about 4 kilobytes.

Claim 66: (Cancelled)

Claim 67:

The ~~method~~software program of claim 59 wherein said resource-constrained device comprises a smart card.

Claim 68:

The ~~method~~software program of claim 59 wherein said resource-constrained device comprises an application-specific integrated circuit (ASIC).

Claim 69:

~~An~~A computer-implemented method for converting and transforming an application software program, ~~comprising:~~ to generate an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions ~~residing on a computer-readable medium~~, said instructions comprising operation codes and operands, said ~~program~~ sequence of instructions operable to be loaded to and executed by a resource-constrained device, said instructions previously converted from at least one class file, the method comprising:

(1) analyzing instructions from said at least one class file to identify (a) plurality of instructions that can be collapsed into a single instruction and (b) instructions including arithmetic operations having 32-bit operands; and

(2) substituting said single instruction for said plurality of instructions that can be collapsed and transforming said 32-bit operands based upon said analyzing by: said instructions comprising

(A) substituting at least one composite instruction for performing an operation on a current object, the execution of said at least one composite instruction, on said resource-constrained device, generating a result functionally equivalent to a result generated by operations resulting from

Art Unit: 2193

~~sequential execution of two or more other instructions so that said~~
~~sequential execution of said two or more other instructions is replaced by~~
~~said execution of said composite instruction in said sequence of~~
~~instructions executed on said resource-constrained device~~ for said plurality
of instructions being collapsed, wherein

(i) a first instruction in said plurality of instructions being
collapsed is any one of a family of instructions, wherein

(a) each instruction in said family is for a different one of a
plurality of data types so that the data type is explicit for
each instruction in said family; and

(ii) a second instruction in said plurality of instructions being
collapsed is not from said family of instructions; and

(B) transforming 32-bit operands for arithmetic operations such that
arithmetic instructions are explicitly defined as using 16-bit operands
wherein upon execution of said arithmetic instructions by a virtual
machine on the resource-constrained device, only 16-bit operands are
loaded onto a stack; and

wherein the instructions formed by substitution and transformation according to (A), (B)
are directly executable by the virtual machine on said resource-constrained device
without any other intermediate stage of reconversion or referencing.

Claim 70:

Art Unit: 2193

The ~~method~~software program of claim 69 wherein said resource-constrained device is based on a 16-bit processor architecture.

Claim 71:

The ~~method~~software program of claim 69 wherein said resource-constrained device is based, on an 8-bit processor architecture.

Claim 72:

The ~~method~~software program of claim 69 wherein said resource-constrained device comprises a random access memory with a capacity of no more than about 64 kilo-bytes.

Claim 73:

The ~~method~~software program of claim 69 wherein said resource-constrained device comprises a random access memory with a capacity of no more than about 4 kilo-bytes.

Claim 74: (Cancelled).

Claim 75:

The ~~method~~software program of claim 69 wherein said resource-constrained device comprises a smart card.

Claim 76:

The ~~method~~software program of claim 69 wherein said resource-constrained device comprises an application-specific integrated circuit (ASIC).

Claim 77:

A resource-constrained device comprising:

Art Unit: 2193

a virtual machine implemented on a microprocessor, said virtual machine configured to execute an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions;

and

a memory, coupled to said virtual machine, for storing having stored therein said an application software program comprising an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions, said instructions comprising operation codes and operands, said sequence of instructions previously converted from at least one class file by a method comprising:

(A) analyzing instructions from said at least one class file and any associated references to a constant pool and arithmetic operations having 32-bit operands, said conversion; and

(B) removing said references to the constant pool and transforming said 32-bit operands based upon said analyzing by:

(i) transforming at least one reference, of at least one of said instructions, to information in a constant pool to data inlined by inlining directly in at least one additional operand or opcode of said at least one of said instructions so that said at least one reference to information in said constant pool is eliminated because accessing said information in any constant pool is unnecessary for said at least one of said instructions; to said at least one instruction, wherein said at least one additional operand specifies a data type;

Art Unit: 2193

(ii) replacing an operation code of another of said instructions, which accesses an entry in said constant pool to obtain a data type that was any one of a plurality of data types, with another operation code selected from a family of operation codes, wherein

(a) each said another operation code in said family is for a different one of said plurality of data types so that the data type is explicit for said each said another operation code in said family; and

(b) said another operation code represents said family for said entry in said constant pool; and

(iii) transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; and

wherein the instructions formed by replacement and transformation according to (i), (ii), (iii) are directly executable by the virtual machine on said resource-constrained device without any other intermediate stage of reconversion or referencing.

~~a virtual machine implemented on a microprocessor—said virtual machine configured to execute said sequence of instructions.~~

Claim 78: (Cancelled)

Art Unit: 2193

Claim 79: (Cancelled)

Claim 87:

A resource-constrained device comprising:

a virtual machine implemented on a microprocessor, in said resource-constrained device,

said virtual machine configured to execute an application software program; and

a memory, in said resource constrained device, ~~for storing~~ having stored therein said an

application software program comprising an object-oriented, verifiable, type-safe and

pointer-safe sequence of instructions, said instructions comprising operation codes and

operands, said instructions previously converted from at least one class file by a method

comprising:

(1) analyzing instructions from said at least one class file to identify (a) plurality of instructions that can be collapsed into a single instruction and (b) instructions including arithmetic operations having 32-bit operands; and

(2) substituting said single instruction for said plurality of instructions that can be collapsed and transforming said 32-bit operands based upon said analyzing by: ;
~~said instructions comprising~~

(A) substituting at least one composite instruction for performing an operation on a current object, ~~the execution of said at least one composite instruction, on said resource-constrained device,~~
~~generating a result functionally equivalent to a result generated by operations resulting from sequential execution of two or more other instructions so that said sequential execution of said two or~~

Art Unit: 2193

~~more other instructions is replaced by said execution of said composite instruction in said sequence of instructions executed on said resource-constrained device.~~ for said plurality of instructions being collapsed, wherein

(i) a first instruction in said plurality of instructions being collapsed is any one of a family of instructions, wherein

(a) each instruction in said family is for a different one of a plurality of data types so that the data type is explicit for each instruction in said family; and

(ii) a second instruction in said plurality of instructions being collapsed is not from said family of instructions; and

(B) transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; and

wherein the instructions formed by substitution and transformation according to (A), (B) are directly executable by the virtual machine on said resource constrained device without any other intermediate stage of reconversion or referencing.

Claim 95:

Art Unit: 2193

A method for using an application software program including an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions, the method comprising:

receiving said software program in a resource-constrained device having a memory and storing said sequence of instructions therein, said instructions comprising operation codes and operands, said sequence of instructions previously converted from at least one class file by a method comprising:

(A) analyzing instructions from said at least one class file and any associated references to a constant pool and arithmetic operations having 32-bit operands,
said conversion; and

(B) removing said references to the constant pool and transforming said 32-bit operands based upon said analyzing by:

(i) transforming at least one reference, of at least one of said instructions, to information in a constant pool to data inlined by inlining directly in at least one additional operand or opcode of said at least one of said instructions so that said at least one reference to information in said constant pool is eliminated because accessing said information in any constant pool is unnecessary for said at least one of said instructions; to said at least one instruction, wherein said at least one additional operand specifies a data type;

(ii) replacing an operation code of another of said instructions, which accesses an entry in said constant pool to obtain a data type

that was any one of a plurality of data types, with another
operation code selected from a family of operation codes, wherein
(a) each said another operation code in said family is for a
different one of said plurality of data types so that the data
type is explicit for said each said another operation code in
said family; and
(b) said another operation code represents said family for
said entry in said constant pool; and
(iii) transforming 32-bit operands for arithmetic operations such
that arithmetic instructions are explicitly defined as using 16-bit
operands wherein upon execution of said arithmetic instructions by
a virtual machine on the resource-constrained device, only 16-bit
operands are loaded onto a stack; and
wherein the instructions formed by replacement and transformation
according to (i), (ii), (iii) are directly executable by the virtual machine on
said resource constrained device without any other intermediate stage of
reconversion or referencing; and
executing said sequence of instructions on said resource-constrained device using said
virtual machine.

Claim 97: (Cancelled)

Claim 99: (Cancelled)

Claim 100: (Cancelled)

Claim 101: (Cancelled)

Claim 106: (Cancelled)

Claim 109:

A method for using an application software program including an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions, the method comprising:

receiving said software program in a resource-constrained device having a memory, said instructions comprising operation codes and operands, said instructions previously converted from at least one class file by a method comprising:

(1) analyzing instructions from said at least one class file to identify (a) plurality of instructions that can be collapsed into a single instruction and (b) instructions including arithmetic operations having 32-bit operands; and

(2) substituting said single instruction for said plurality of instructions that can be collapsed and transforming said 32-bit operands based upon said analyzing by: ;
~~said instructions comprising~~

(A) substituting at least one composite instruction for performing an operation on a current object, ~~the execution of said at least one composite instruction, on said resource-constrained device,~~
~~generating a result functionally equivalent to a result generated by operations resulting from sequential execution of two or more other instructions so that said sequential execution of said two or more other instructions is replaced by said execution of said composite instruction in said sequence of instructions executed on~~

~~said resource constrained device.~~ for said plurality of instructions being collapsed, wherein

(i) a first instruction in said plurality of instructions being collapsed is any one of a family of instructions, wherein

(a) each instruction in said family is for a different one of a plurality of data types so that the data type is explicit for each instruction in said family; and

(ii) a second instruction in said plurality of instructions being collapsed is not from said family of instructions; and

(B) transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; and

wherein the instructions formed by substitution and transformation according to (A), (B) are directly executable by the virtual machine on said resource constrained device without any other intermediate stage of reconversion or referencing; and

executing said sequence of instructions on said resource-constrained device.

Claim 113: (Cancelled)

Claim 114: (Cancelled)

Claim 115: (Cancelled)

Claim 120: (Cancelled)

Claim 123:

An apparatus for using an application software program including an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions, the apparatus comprising:

means for receiving said software program in a resource-constrained device having a memory, said instructions comprising operation codes and operands, said sequence of instructions previously converted from at least one class file by a method comprising:

(A) analyzing instructions from said at least one class file and any associated references to a constant pool and arithmetic operations having 32-bit operands, said conversion; and

(B) removing said references to the constant pool and transforming said 32-bit operands based upon said analyzing by:

(i) transforming at least one reference, of at least one of said instructions, to information in a constant pool to data inlined by inlining directly in at least one additional operand or opcode of said at least one of said instructions so that said at least one reference to information in said constant pool is eliminated because accessing said information in any constant pool is unnecessary for said at least one of said instructions; to said at least one instruction, wherein said at least one additional operand specifies a data type;

Art Unit: 2193

(ii) replacing an operation code of another of said instructions, which accesses an entry in said constant pool to obtain a data type that was any one of a plurality of data types, with another operation code selected from a family of operation codes, wherein

(a) each said another operation code in said family is for a different one of said plurality of data types so that the data type is explicit for said each said another operation code in said family; and

(b) said another operation code represents said family for said entry in said constant pool and

(iii) transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; and

wherein the instructions formed by replacement and transformation according to (i), (ii), (iii) are directly executable by the virtual machine on said resource constrained device without any other intermediate stage of reconversion or referencing; and

means for executing said sequence of instructions on said resource-constrained device.

Claim 127: (Cancelled)

Claim 128: (Cancelled)

Claim 129: (Cancelled)

Claim 134: (Cancelled)

Claim 137:

An apparatus for using an application software program including an object-oriented, verifiable, type-safe and pointer-safe sequence of instructions, the apparatus comprising:

means for receiving said software program in a resource-constrained device having a memory, said instructions comprising operation codes and operands, said instructions previously converted from at least one class file by a method comprising:

(1) analyzing instructions from said at least one class file to identify (a) plurality of instructions that can be collapsed into a single instruction and (b) instructions including arithmetic operations having 32-bit operands; and, said conversion

(2) substituting said single instruction for said plurality of instructions that can be collapsed and transforming said 32-bit operands based upon said analyzing by:
~~said instructions~~

(A) substituting at least one composite instruction for performing an operation on a current object, ~~the execution of said at least one composite instruction, on said resource-constrained device, generating a result functionally equivalent to a result generated by operations resulting from sequential execution of two or more other instructions so that said sequential execution of said two or more other instructions is replaced by said execution of said composite instruction in said sequence of instructions executed on~~

said resource-constrained device for said plurality of instructions
being collapsed, wherein

(i) a first instruction in said plurality of instructions being
collapsed is any one of a family of instructions, wherein

(a) each instruction in said family is for a different
one of a plurality of data types so that the data type
is explicit for each instruction in said family; and

(ii) a second instruction in said plurality of instructions
being collapsed is not from said family of instructions; and

(B) transforming 32-bit operands for arithmetic operations such
that arithmetic instructions are explicitly defined as using 16-bit
operands wherein upon execution of said arithmetic instructions by
a virtual machine on the resource-constrained device, only 16-bit
operands are loaded onto a stack; and

wherein the instructions formed by substitution and transformation
according to (A), (B) are directly executable by the virtual machine on
said resource constrained device without any other intermediate stage of
reconversion or referencing; and

means for executing said sequence of instructions on said resource-constrained device.

Claim 141: (Cancelled)

Claim 142: (Cancelled)

Claim 143: (Cancelled)

Claim 148: (Cancelled)

II) In the Specifications: **CROSS REFERENCE TO RELATED APPLICATIONS**
(middle, pg. 1)

Amend as follows:

U.S. Patent Application Serial No. 09/243,107, filed February 2, 1999, in the names of inventors Judith E. Schwabe and Joshua Susser, entitled "Zero Overhead Exception Handling", Attorney Docket No. SUN-P3732, commonly assigned herewith, now issued as U.S. Patent Serial No 6,848,111;

U.S. Patent Application Serial No. 09/243,108, filed February 2, 1999, in the names of inventors Judith E. Schwabe and Joshua Susser, entitled "Token-Based Linking", Attorney Docket No. SUN-P3730, commonly assigned herewith, now issued as U.S. Patent Serial No 6,880,155;

EXAMINER'S STATEMENT OF REASONS FOR ALLOWANCE

3. Claims 59, 62-65, 67-73, 75-77, 80-96, 98, 102-105, 107-112, 116-119, 121-126, 130-133, 135-140, 144-147, 149-150 are allowed.

The following is an examiner's statement of reasons for allowance.

The prior art taken separately or jointly does not suggest or teach the following features.

A method for support of runtime execution by the virtual machine on a resource-constrained device by transforming program instructions to be executed by the device so to eliminate all references to a constant pool by the runtime virtual machine, the instructions being

Art Unit: 2193

previously converted from a class file and having operands and opcodes being analyzed with associated references thereof to a constant pool; and based on such analysis, the transformation method comprising:

(i) replacing an operation code of an instruction that accesses an entry in said constant pool to obtain a data type, with another operation code selected to represent a family for an entry in said constant pool, wherein each said another operation code is for a different one of said plurality of data types, such that the data type is explicit for said each said another operation code;

(ii) directly inlining an additional operand in an instruction that references said constant pool such that the added operand specifies a data type; as recited in claims 59, 77, 95 and 123;

(iii) identifying a plurality of collapsible instructions and forming a composite instruction based thereupon, such that a first instruction in said plurality of instructions being collapsed is any one of a family of instructions, wherein each instruction in said family is for a different one of a plurality of data types so that the data type is explicit for each instruction in said family; and a second instruction in said plurality of instructions being collapsed is not from said family of instructions; as recited in claims 69, 87, 109, and 137;

(iv) for identified arithmetic operation instructions, transforming 32-bit operands for arithmetic operations such that arithmetic instructions are explicitly defined as using 16-bit operands wherein upon execution of said arithmetic instructions by a virtual machine on the resource-constrained device, only 16-bit operands are loaded onto a stack; as recited in claims 59, 69, 77, 87, 95, 109, 123 and 137.

Wilkinson, USPN: 6308317, discloses loading instructions from a CAP file into a resource -constraint Smart Card for execution; parsing class file instructions and readjusting references thereof to a constant pool with replacement of references with their updated values and/or equivalent opcodes; and renumbering the original instructions with new sequence number in the JVM card; but Wilkinson fails to teach or suggest the combined teaching of (i), (ii) and (iv); or (iii) and (iv), as set forth above.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.


The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Art Unit: 2193

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
July 21, 2006



KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100